Perancangan dan Implementasi *Payment Gateway* dengan metode *Concurrency* untuk Transaksi Nontunai

Yoyok Gamaliel [#1], Sinung Suakanto[#2], Andreas[#3]

[#]*Departemen Teknologi Informasi, Institut Teknologi Harapan Bangsa*
Jl. Dipatiukur no. 80-84, Bandung, Jawa Barat, Indonesia

[1]yoyok@ithb.ac.id

[2]sinung@ithb.ac.id

[3]ndreaja16@gmail.com

Abstract*— Nowadays, the use of technology in the non-cash transaction system has started to shift the system of cash transaction. In line with this situation, the issue of concurrency, the system's ability to handle the large amount of transactions at the same time, is emerging. This matter may result in lost updates, uncommitted dependencies, and inconsistent analysis. Therefore, it is required to build a payment gateway system that is able to handle this issue. In addition, the system must apply the standard format for non-cash transactions. This paper will design a payment gateway system using concurrency mechanism by applying Akka framework and ISO 8583 as a message format for non-cash transaction. The implementation of the payment gateway system will enable the system to handle concurrency and to use standard format transaction. The payment gateway system is tested with 2 parameters, namely functionality and concurrency.

***Keywords*— Concurrency, Non-cash transactions, payment gateway, framework* Akka, ISO8583**

Abstrak— Sistem transaksi non-tunai yang berbasis teknologi kini telah mulai menggantikan system transaksi tunai. Berkaitan dengan hal tersebut muncul isu tentang *concurrency*, yaitu kemampuan system dalam menangani jumlah transaksi yang banyak dalam waktu bersamaan. Adanya isu tersebut dapat mengakibatkan *lost update*, *uncommitted dependency*, dan *inconsistent analysis*. Oleh karena itu diperlukan sebuah sistem *payment gateway* yang mampu menangani isu tersebut. Selain itu, sistem tersebut harus menerapkan standar format untuk transaksi nontunai. Makalah ini akan merancang sebuah sistem *payment gateway* yang menggunakan mekanisme *concurrency* dengan menggunakan *framework* Akka dan ISO 8583 sebagai format pesan untuk transaksi nontunai. Implementasi dari sistem *payment gateway* tersebut akan membuat sistem tersebut mampu menangani concurrency dan menggunakan standard format data transaksi. *Sistem payment gateway* ini diuji dengan 2 parameter, yaitu fungsionalitas dan *concurrency*.

Kata kunci— *concurrency*, transaksi nontunai, *payment gateway*, *framework* Akka, ISO8583.

I. PENDAHULUAN

E-payment merupakan sistem pembayaran nontunai melalui jaringan internet. Pada saat ini masyarakat dapat melakukan transaksi tanpa menggunakan uang tunai seperti kartu kredit, kartu debit, dan lain-lain. Transaksi nontunai telah dilakukan dalam volume yang besar. Sejak diluncurkannya Gerakan Nasional Non Tunai (GNNT) pada Agustus 2014, transaksi nontunai telah mencapai Rp 4,3 triliun pada Oktober 2015 dan volume transaksi sekitar 450 juta kali transaksi[3].

Hasil percobaan[9] berdasarkan model *single-threaded* menunjukkan bahwa waktu rata-rata total untuk proses pembayaran dan proses pada *payment gateway* lebih pendek dibandingkan dengan model *multiple-threaded*. Alasan utama dikarenakan adanya *resource conflict* untuk model *multiple-threaded* ketika *concurrent processes* mengakses *Payment Gateway* yang hanya memiliki akses ke satu *server*[9]. Hal ini menjadi masalah dalam motode *concurrency*. Masalah lainnya yang dapat muncul adalah *lost update*, *uncommitted data*, dan *inconsistent retrievals*[2]. Oleh karena itu, diperlukan suatu sistem yang memiliki kemampuan dalam menangani *concurrency* tersebut.

Framework Akka merupakan sebuah model pemrogramaan dengan mekanisme *concurrency*. Akka telah banyak diadopsi oleh banyak organisasi besar untuk digunakan dalam berbagai industri seperti investasi, bank, *merchant*, retail, media sosial, *health care* dan analisis data[8].

Transaksi nontunai memerlukan standar format data dalam melakukan setiap transaksi. Standar tersebut antara lain ISO 8583 dan ISO 20022. Kedua standar tersebut digunakan dalam melakukan transaksi nontunai. ISO 8583 digunakan untuk transaksi nontunai dengan menggunakan kartu, sedangkan ISO 20022 digunakan untuk transaksi nontunai antar lembaga keuangan.

Tujuan dari makalah ini adalah bagaimana merancang sistem *payment gateway* yang mampu menangani *concurrency* dan menerapkan format pesan ISO 8583 untuk transaksi nontunai. Manfaat dari makalah ini adalah dapat membantu pihak *merchant* dalam melakukan banyak transaksi ke banyak *server* dengan menggunakan format data transaksi yang sama dan didukung oleh mekanisme *concurrency*.

II. KAJIAN PUSTAKA

Pada bab ini akan diuraikan mengenai transaksi nontunai, ISO 8583, *payment gateway*, *concurrency*, dan *framework* Akka.

A. Transaksi Non Tunai

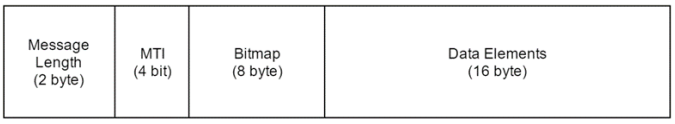
Transaksi nontunai merupakan transaksi tanpa menggunakan uang secara langsung (*cash*). Pembayaran nontunai melibatkan jasa perbankan dalam penerapannya. Bank sebagaimana fungsinya yaitu menghimpun dana masyarakat, tentunya memberikan dan menyediakan jasa dalam lalu lintas pembayaran bagi nasabahnya. Jasa dalam lalu lintas pembayaran tersebut antara lain melalui cek, transfer dana dari suatu rekening simpanan ke rekening simpanan lainnya pada bank yang sama atau pada bank yang berbeda, melalui kartu kredit, kartu debet, dan lain-lain[1].

Berdasarkan referensi [7] terdapat beberapa jenis transaksi nontunai, seperti :

- Cek
- Kartu kredit
- Kartu debet
- *Electronic money*

B. ISO 8583

ISO 8583 merupakan sebuah format pesan dan aliran komunikasi yang dapat berjalan pada sistem yang berbeda sehingga dapat melakukan pertukaran transaksi, baik itu *request* maupun *response*. ISO 8583 digunakan sebagai standar pertukaran pesan transaksi antara *acquirers* dengan *card issuers*. Transaksi yang dilakukan pada ATM dan mesin pembayaran *merchant* menggunakan format pesan ISO 8583 untuk pertukaran *key exchange*, total rekonsiliasi dan keperluan administrasi.



Gambar 1 Komponen Penyusun Format Pesan ISO 8583

Gambar 1 menunjukkan komponen penyusun format pesan ISO 8583, yaitu [5]:

- *Message Length*
  Message length memuat informasi mengenai total lebar sebuah pesan. Lebar data dari *message length* adalah 2 byte.
- *MessageType Indicator* (MTI)
  MTI memuat informasi mengenai versi ISO-8583, *message class*, *message function*, dan *message origin*. Lebar data dari MTI adalah 4 bit.
- *Bitmap*
  Bitmap memuat informasi mengenai pesan yang menunjukkan *data elements* yang digunakan. Lebar data dari *bitmap* adalah 8 byte.
- *Data Elements*
  Data elements memuat informasi transaksi. Lebar data dari *data elements* adalah 16 byte.

Berbagai jenis transaksi seperti melakukan cek informasi saldo, *authorization*, verifikasi, pembayaran melalui *point of sale*, penarikan tunai, transfer, dan pengembalian memiliki formatnya masing-masing, bergantung pada *data element* nomor 24 dan MTI. MTI untuk cek informasi saldo, *authorization* dan verifikasi adalah 100 (*request*) dan 110 (*response*). Sedangkan MTI untuk pembayaran melalui *point of sale*, penarikan tunai , pembayaran, transfer, dan pengembalian adalah 200 (*request*) dan 210 (*response*). Tabel 1 berisi informasi mengenai format yang digunakan untuk melakukan transaksi.

TABEL I

FORMAT TRANSAKSI ISO 8583

Field	Description	Req	Resp
MTI	Message Type Indicator	100	100
2	DE-002 Primary Account Number (PAN)	07	16
3	DE-003 Processing Code	M	27
4	DE-004 Amount transaction	26	26
7	DE-007 Date and time transmission	30	30
11	DE-011 System Trace Audit Number (STAN)	M	ME
12	DE-012 Date and time local transaction	M	ME
13	DE-013 Data effective	02	
14	DE-014 Expiration date	02	
18	DE-018 Message *Error* Indicator		36
21	DE-021 Transaction life cycle identification data	33	33
22	DE-022 Point-of-Service data code	M	
23	DE-023 jCard Sequence Number	02	
24	DE-024 Function Code	M	
26	DE-026 *Merchant* Category Code	M	
30	DE-030 Amount original	O	O
32	DE-032 Acquiring institution identification code	O	O
35	DE-035 Track 2 data	06	
38	DE-038 Approval Code		31
39	DE-039 Result Code		M
41	DE-041 Card Acceptor Terminal ID	15	16
42	DE-042 Card Acceptor Identification Code	15	16
43	DE-043 Card acceptor name/location	O	O
45	DE-045 Track 1 data	06	
46	DE-046 Amount fees	O	O
49	DE-049 Verification Data	45	45
59	DE-059 Transport Data	O	16
63	DE-063 Display message	O	ME
111	DE-111 Discretionary handback data	O	ME

C. Payment Gateway

Payment gateway merupakan layanan yang mengotorisasi pembayaran dalam *e-business* dan *online retails*. *Payment gateway* setara dengan POS (*point-of-sale*) yang berada pada *outlet* atau *merchant*. Umumnya *payment gateway* memiliki 2 komponen, yaitu [6]:

- *virtual terminal* yang mengizinkan sebuah *merchant* untuk menjaga keamanan *login* dan *key* pada nomor kartu kredit atau
- *website shopping-cart* yang terhubung dengan *gateway* melalui API, sehingga memungkinkan *real time processing* dari *website merchant* tersebut.

D. Concurrency

Concurrency merupakan salah satu isu pada sistem transaksi nontunai karena dalam transaksi nontunai banyak transaksi yang terjadi pada waktu yang bersamaan. *Concurrency* dapat menimbulkan beberapa masalah, diantaranya [2]:

- *Lost update*
  Masalah yang muncul ketika dua proses melakukan perubahan data di waktu yang sama sehingga perubahan data pada salah satu proses mengalami *overwritten* oleh proses yang lain.
- *Uncommitted data*
  Masalah yang muncul ketika terdapat satu buah proses yang gagal melakukan perubahan data sehingga terjadi *rolled back.* Di waktu yang bersamaan, proses yang lain mengakses *uncommitted* data tersebut sehingga data yang diperoleh akan salah.
- *Inconsistents retrievals*
  Inconsistents retrievals terjadi ketika sebuah transaksi membaca beberap nilai data dari database, tetapi ada transaksi kedua yang merubah beberapa nilai data tersebut pada saat transaksi pertama masih berlangsung.

E. Framework Akka

Framework Akka merupakan sebuah *toolkit* bersifat *open source* yang menyediakan kemudahan, *concurrent*, dan aplikasi distribusi. Adanya *framework* Akka dapat membuat sebuah aplikasi yang dapat berjalan pada *cloud* atau dibanyak perangkat dan efisien dalam pemanfaatan kapasitas komputasi.

Inti dari Akka disebut *actor*[8]. *Actor* merupakan sebuah obyek yang memiliki *state* dan *behavior*. *Actor* berkomunikasi dengan melakukan pertukaran pesan dimana pesan tersebut disimpan pada kotak pesan penerima [8]. *Actor* merupakan perwujudan dari transaksi *single-thread* yang terjadi secara *concurrent* dan *asynchronous* secara menyeluruh pada semua sumber yang tersedia di sebuah aplikasi[4].

III. Analisis

Analisis sistem transaksi nontunai dilakukan dengan melihat kebutuhan sistem transaksi nontunai, spesifik sistem transaksi nontunai, dan perancangan sistem transaksi nontunai dengan *payment gateway*.

A. Analisis Kebutuhan Sistem Transaksi Nontunai

Pada umumnya sistem transaksi nontunai konvensional melibatkan pihak *merchant* dengan *server*, dimana *merchant* mengirimkan *request* data transaksi langsung ke *server*. Setelah itu, *server* mengolah data transaksi dan mengirimkan *response* kepada *merchant*. Akan tetapi hal tersebut dapat menurunkan kerja sebuah *server* apabila *server* menangani *request* dari banyak *merchant*. Selain itu, sebuah *merchant* yang terhubung ke banyak *server* dapat mengirimkan pesan dengan format yang berbeda-beda. Dari sistem konvensional dapat dilihat bahwa diperlukan sebuah teknologi yang menghubungkan antara *server* dengan *merchant*. Teknologi tersebut harus dapat menangani jumlah transaksi yang banyak dalam waktu yang bersamaan dan menggunakan satu format data transaksi yang sama.

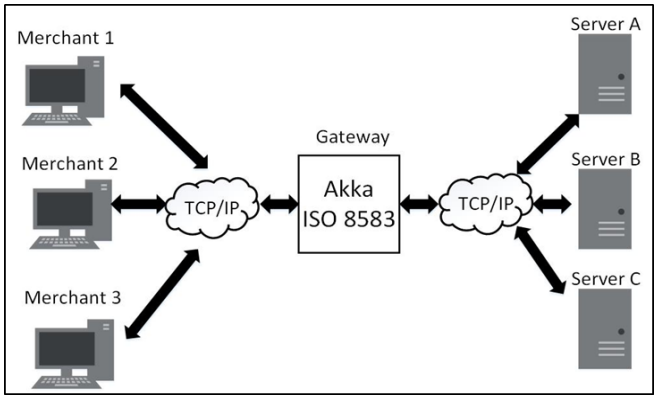
B. Spesifikasi Sistem Transaksi Nontunai

Teknologi yang mampu mengatasi kekurangan pada sistem transaksi konvensional adalah *payment gateway*. *Payment gateway* berfungsi untuk menangani jumlah transaksi yang banyak dari beberapa *merchant* dalam waktu yang bersamaan dengan menggunakan format data transaksi yang sama. *Payment gateway* menggunakan *framework* Akka yang memiliki kemampuan *concurrency* dan menggunakan ISO 8583 sebagai format pesan transaksi.

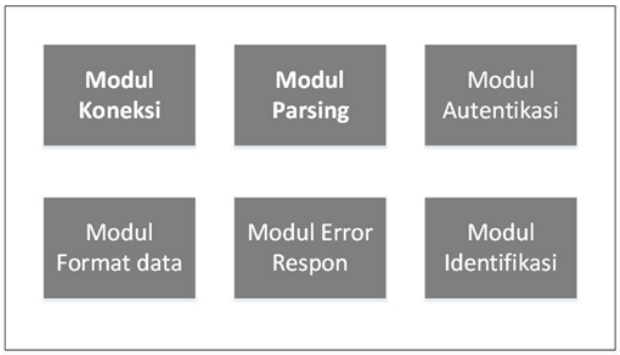
C. Perancangan Sistem Transaksi Nontunai dengan Payment Gateway

Sistem transaksi nontunai dengan *payment gateway* terdiri 3 bagian, yaitu *merchant*, *payment gateway*, dan *server*. Arsitektur sistem dapat dilihat pada Gambar 2.

Merchant berfungsi sebagai tempat untuk *user* melakukan transaksi. *Payment gateway* menghubungkan antara *merchant* dengan *server*. *Server* berfungsi untuk memproses data transaksi dari *mechant*.



Gambar 2 Arsitektur Payment Gateway



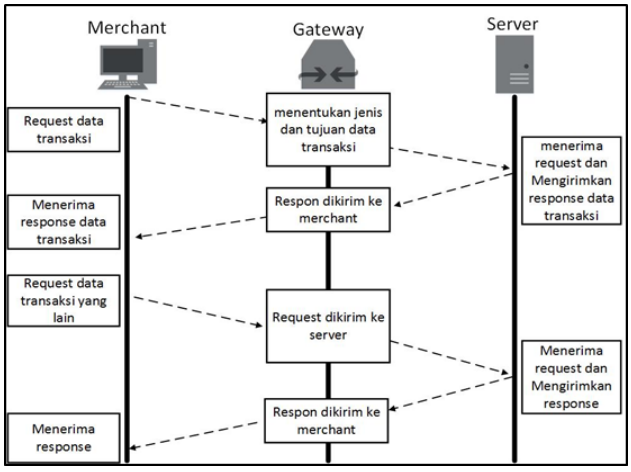
Gambar 3 Modul Payment Gateway

Perancangan *payment gateway* ini terdiri dari beberapa modul. Modul-modul tersebut dapat dilihat pada Gambar 3.

Fungsi-fungsi dari setiap modul yaitu :

- Modul Koneksi
  Fungsi dari modul koneksi adalah menghubungkan *merchant* dengan *server*. Data yang diterima dan dikirim oleh *payment gateway* diatur oleh modul koneksi.
- Modul *Parsing*
  Fungsi dari modul *parsing* adalah untuk melakukan pembagian data agar mendapatkan data yang dibutuhkan.
- Modul Autentikasi
  Fungsi dari modul autentikasi adalah untuk memberikan izin kepada *merchant* untuk mengirimkan data transaksi.
- Modul Format Data
  Fungsi dari modul format data adalah untuk menyesuaikan format data transaksi dengan format yang digunakan *merchant* dan *server*.
- Modul *Error Response*
  Fungsi dari modul *error response* adalah untuk memberikan *response* kepada *merchant* jika terjadi kesalahan atau kegagalan transaksi.
- Modul Identifikasi
  Fungsi dari modul identifikasi adalah untuk mengidentifikasi data transaksi tersebut sehingga dapat menentukan *server* atau *merchant* yang dituju.

Gambar 4 menunjukkan alur transaksi yang terjadi antara *merchant* dengan *server* melalui *payment gateway*.



Gambar 4 Alur Payment Gateway

IV. IMPLEMENTASI

Implementasi sistem *payment gateway* menjelaskan mengenai proses implementasi perangkat lunak, modul, data transaksi, dan program pada *payment gateway*.

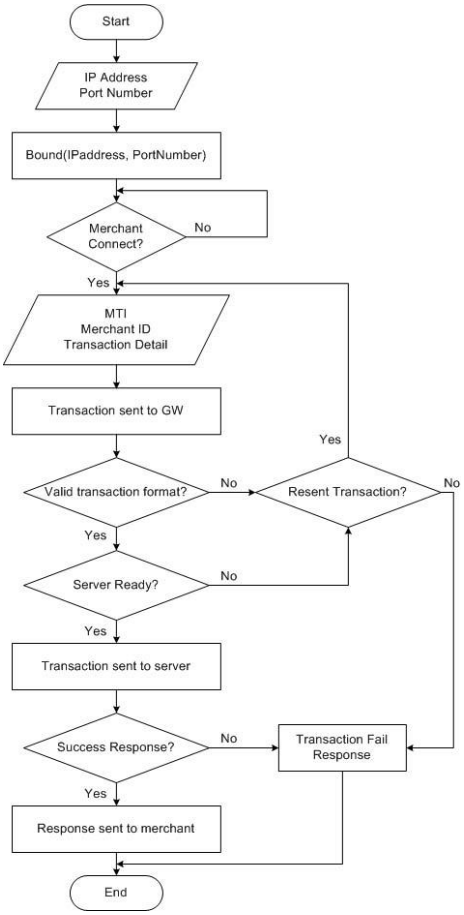
A. *Implementasi Perangkat Lunak*

Perangkat lunak digunakan untuk membantu pengembangan sistem *payment gateway*. Perangkat lunak yang digunakan adalah *framework* Akka dan ISO8583. *Framework* Akka ini digunakan untuk menangani jumlah data transaksi yang banyak dalam waktu bersamaan. ISO 8583 digunakan untuk protokol dan format pesan transaksi nontunai. Diagram alur pada Gambar 5 menunjukkan implementasi perangkat lunak proses transaksi pada payment gateway.

B. *Implementasi Data Transaksi*

Data transaksi yang dibutuhkan oleh sistem *payment gateway* sebagai berikut:

- MTI (*Message Type Identifier*). MTI digunakan untuk mengetahui jenis transaksi
- *ID*. *ID* digunakan untuk menentukan tujuan data transaksi dari *merchant* ke *server*.
- *Merchant Code*. *Merchant code* digunakan untuk menentukan tujuan data transaksi dari *server* ke *merchant*.

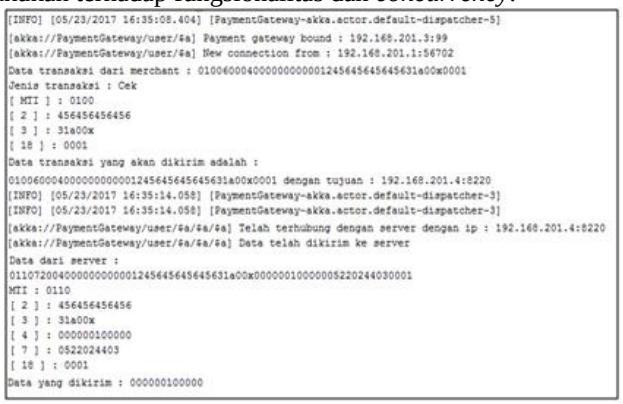


Gambar 5 Proses Transaksi Payment Gateway

V. PENGUJIAN

Proses yang terjadi pada *payment gateway* ketika menerima dan mengirimkan pesan dari *merchant* ke *server* dapat dilihat pada Gambar 6.

Pengujian dilakukan pada implementasi sistem yang telah dilakukan. Pengujian yang dilakukan untuk menjadi tolak ukur dari *payment gateway*. Pengujian dilakukan dengan menggunakan jaringan internet yang ada di kampus. Pengujian dilakukan terhadap fungsionalitas dan *concurrency*.



Gambar 6 Proses Data Transaksi pada Payment Gateway

A. Fungsionalitas

Berdasarkan fungsionalitas dari *payment gateway*, pengujian dibagi menjadi 4 yaitu :

- *Payment gateway* mampu melakukan *request* dan *response*. Hasil pengujian dapat dilihat pada Tabel II.

TABEL II

PENGUJIAN FUNGSIONALITAS *REQUEST* DAN *RESPONSE*

Request			Response		
Field	**Digit**	**Nilai**	**Field**	**Digit**	**Nilai**
MTI	0-3	0100	MTI	0-3	0110
1 (bitmap)	4-21	6000400000000012	1 (bitmap)	4-21	7000400000000012
2 (ID)	22-33	456456456456	2 (ID)	22-33	456456456456
3 (Kode Proses)	34-39	31a00x	3 (kode proses)	34-39	31a00x
18 (Merchant ID)	40-43	0001	4 (amount)	40-51	000000055000
			18 (Merchant ID)	52-55	0001
0100600040000000000012456456456456631a00x0001			0110700040000000000012456456456456631a00x0000000550000001		
Waktu rata-rata / transaksi			0.05265s		

- *Payment gateway* menerima *request* data transaksi yang tidak sesuai format. Hasil pengujian dapat dilihat pada Tabel III.

TABEL III

PENGUJIAN FUNGSIONALITAS *REQUEST* TIDAK SESUAI FORMAT

Request Merchant	**Response Gateway**
120070004000000000000745646230 1a00x0000000010002341 MTI Bitmap ID Kode Proses Amount Kode Merchant	30

- *Payment gateway* menerima duplikat transaksi. Hasil pengujian dapat dilihat pada Tabel IV.

TABEL IV

PENGUJIAN FUNGSIONALITAS DUPLIKAT TRANSAKSI

Kondisi	**Request**	**Response**
Request 1	0100600040000000 0012456456456456 01 a00x0001	0110700040000000 00124564564564560 1a00x0000000550000001
Request 2 (duplikat)	0100600040000000 0012456456456456 01 a00x0001	94

B. Concurrency

Pengujian sistem dalam menangani *concurrency* dilakukan untuk mengetahui kemampuan sistem dalam menangani banyak transaksi dari beberapa *merchant*. Terdapat 2 skema pengujian sistem dalam menangani *concurrency*, yaitu :

- Skema A (*single merchant*) adalah *payment gateway* menerima data transaksi dari 1 *merchant*. Skema A ini memiliki 3 kondisi, yang terdiri dari :
  - Kondisi 1: *payment gateway* menangani 100 transaksi dari 1 *merchant*.
  - Kondisi 2: *payment gateway* menangani 300 transaksi dari 1 *merchant*.
- Skema B (*multiple merchant*) adalah *payment gateway* menerima data transaksi dari 2 *merchant*. Skema B ini memiliki 3 kondisi, yang terdiri dari :
  - Kondisi 1: *payment gateway* menangani 100 transaksi dari setiap *merchant*.
  - Kondisi 2: *payment gateway* menangani 300 transaksi dari setiap *merchant*.

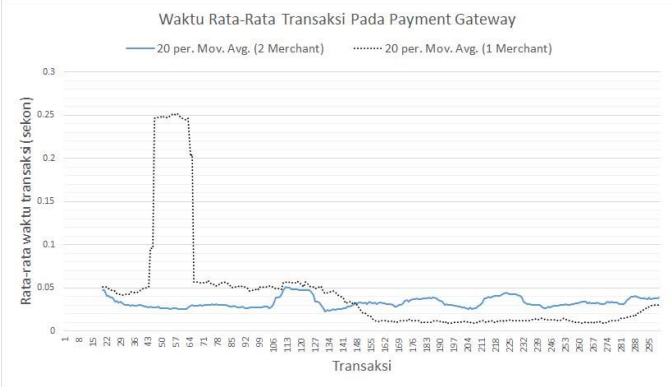
Hasil pengujian dapat dilihat pada Tabel V.

TABEL V

PENGUJIAN *CONCURRENCY*

Jumlah Merchant	Jumlah Transaksi / merchant	Waktu transaksi maksimum	Waktu transaksi minimum	waktu rata-rata/ transaksi
1	100	0.75s	0.015s	0.05265s
1	300	0.187s	0.015s	0.033587 s
2	100	0.219s	0.015s	0.04171s
2	300	3.031s	<0.01s	0.043857 s

Gambar 7 menunjukkan hasil perbandingan waktu rata-rata proses transaksi yang dilakukan oleh satu *merchant* dan dua *merchant*.



Gambar 7 Waktu Rata-Rata Transaksi Payment Gateway

Pada proses transaksi 1 *merchant*, rata-rata waktu yang diperlukan untuk 20 transaksi berada pada keadaan saturasi 0.01 sekon sampai 0.02 sekon. Pada proses transaksi 2 *merchant*, untuk jumlah transaksi yang sama, dicapai keadaan saturasi pada 0,02 sekon sampai 0,05 sekon, dimana terjadi penambahan waktu proses transaksi 0,02 sampai 0,03 sekon.

Rangkuman dari seluruh hasil pengujian fungsionalitas dan *concurrency* dapat dilihat pada Tabel VI.

TABEL VI

KESIMPULAN PENGUJIAN FUNGSIONALITAS DAN *CONCURRENCY*

Skema	Hasil yang diharapkan	Hasil pengamatan	Kesimpulan
Pemrosesan *request* dan *response*	Dapat mengolah data transaksi dari *merchant* dan *server* dan mengirimkan-nya.	*Payment gateway* dapat mengolah dan mengirimkan data transaksi ke *merchant* dan *server*.	[X] OK [ ] Not OK
Pemrosesan data transaksi yang salah	Data transaksi ditolak	*Payment gateway* memberikan *error response code*	[X] OK [ ] Not OK
Pemrosesan duplikat transaksi	Data transaksi duplikat ditolak	*Payment gateway* mengolah *request* pertama dan memberikan *error response* untuk duplikat transaksi	[X] OK [ ] Not OK
Server tidak aktif	Data transaksi ditolak	*Payment gateway* memberikan *No Response Code* ke *merchant*	[X] OK [ ] Not OK
Concurrency	Dapat mengolah data transaksi dari lebih dari 1 *merchant*	*Payment gateway* dapat mengolah data transaksi dari 2 *merchant* dalam waktu bersamaan.	[X] OK [ ] Not OK

VI. KESIMPULAN

Payment gateway yang dirancang dengan menggunkaan *framework* Akka memiliki kemampuan *concurrency*, dimana transaksi yang terjadi dari beberapa *merchant* yang terjadi secara bersamaan dapat diproses dan diotorisasi. Pada pengujian *concurrency* dengan multi *merchant* diperoleh waktu minimum adalah kurang dari 0,01 sekon, dan waktu maksimum adalah 3,031 sekon. Waktu rata-rata transaksi pada satu *merchant* lebih pendek dibandingkan dengan waktu rata-rata dua *merchant*, dimana terjadi perbedaan 0.02 sekon sampai 0.03 sekon. Dari segi fungsionalitas sistem *payment gateway* mampu menerima dan mengirimkan *request* dari *merchant* serta *response* dari *server* dengan waktu rata-rata 0.052 sekon. Selain itu, *payment gateway* mampu menangani kesalahan atau kegagalan transaksi dengan mengirimkan *response code* kepada *merchant*.

Untuk pengembangan ke arah yang lebih baik perlu dilakukan untuk meningkatkan keamanan dan konektivitas. Adapun saran-saran terhadap pengembangan sistem *payment gateway* ini sebagai berikut :

- Menambahkan sisi keamanan dari *payment gateway* dengan mengenkripsi data transaksi.
- *Concurrency* diuji tingkat saturasinya terhadap jumlah transaksi dan jumlah server.

DAFTAR REFERNSI

[1] Ditriano dan Rayna. (2015, November). "Analisis penggunaan transaksi non-tunai di kalangan pengusaha kota medan.". [On-line]. Tersedia: repository.usu.ac.id [Oktober. 30, 2016].

[2] EduGrabs. (2015, September. 2). Concurrency problems in transaction. [On-line]. Available: www.edugrabs.com [October. 12, 2016].

[3] Gianie. 2015. Menjadi masyarakat nontunai. [On-line]. Available: www.print.kompas.com [October. 10, 2016].

[4] J. Allen. (2013, August). Effective Akka. [On-line]. Available: www.it-ebooks.info [September. 28, 2016].

[5] jPOS. jPOS Common Message Format (n.d.). [On-line]. Available: www.jpos.org [October. 30, 2016].

[6] K. Jamdaade and H. Champaneri. (2015, June). "Secured electronic payment gateway." International Journal of Technology Enhancements and Emerging Engineering Research. [On-line]. 3(6). Available: www.ijteee.org [October. 12, 2016].

[7] S. Hidayati, I. Nuryanti, dkk. "Operasional E-Money". [On-line]. Tersedia: http://www.bi.go.id/id/publikasi/jurnal-ekonomi/Default.aspx [November. 10, 2016]

[8] Typesafe Inc,. "Akka Scala Documentation". [On-line]. Tersedia: http://doc.akka.io/docs/akka/2.4.2/AkkaScala.pdf [Februari. 16, 2016]

[9] S. Pattnaik, P.R. Ghosh, A.K. Bharti, "The Working Model of An E-Payment System," Journal of Theoretical And Applied Information Technology", March 2010, vol. 13, pp. 10-13

Yoyok Gamaliel, kelahiran Ciamis tahun 1974 dan memperoleh gelar Sarjana Teknik dari Universitas Kristen Satya Wacana, dan Master of Engineering dari University of South Australia. Minat penelitian pada analisis data serta pemodelan dan simulasi sistem. Saat ini aktif sebagai staf pengajar di Departemen Teknologi Informasi Institut Teknologi Harapan Bangsa.

Sinung Suakanto, kelahiran Klaten tahun 1982 dan memperoleh gelar Sarjana Teknik dari Teknik Elektro ITB. Melanjutkan pendidikan doktoral di bidang Teknik Elektro dalam bidang jaringan komunikasi juga di ITB. Minat penelitian pada bidang jaringan sensor, cloud computing, serta teknologi informasi. Saat ini aktif sebagai staf pengajar di Departemen Teknologi Informasi Institut Teknologi Harapan Bangsa.

Andreas, kelahiran Bandung, Jawa Barat 1994, menyelesaikan S1 di Jurusan Sistem Komputer ITHB pada Agustus 2017.